CS575 Parallel Computing Project#2
OpenMP: Static vs Dynamic and Small vs. Large Chunksize

Shangjia Dong

[1]School of Civil and Construction Engineering, Oregon State University

dongs@oregonstate.edu

1.

Question: Tell what machine you ran this on.

I ran this on rabbit.

- 2 E5-2630 Xeon Processors
- ArchitectureArchitecture: x86_64
- CPU op-mode(s): 32-bit, 64-bit
- CPU(s): 32
- Thread(s) per core: 2
- Core(s) per socket: 8
- Socket(s): 2
- 64 GB of memory
- 2 TB of disk

2.

Question: Create a table with your results.

Table 1. Static and dynamic scheduling and chunksize of 1 and 4096 results comparison: rabbit (unit: megamults per second)

	static-1	static-4096	dynamic-1	dynamic-4096
1	290.29	290.3	290.24	290.3
2	579.44	551.85	579.45	553.88
4	1123.62	961.2	1156.58	809.55
6	1631.19	1213.76	1504.78	1046.49
8	2101.8	1363.36	1905.47	1191.92
10	2308.23	1527.23	2357.4	1278.93
12	2939.9	1698.94	2830.71	1419.25
14	3300.2	1689.95	3302.42	1593.83
16	3770.36	1704.42	3773.72	1723.75
24	4622.52	1668.58	5059.06	1697.26

3.

Question: Draw a graph. The X axis will be the number of threads. The Y axis will be the performance in whatever units you sensibly choose. On the same graph, plot 4 curves.

- static, 1
- static, 4096
- dynamic, 1
- dynamic, 4096

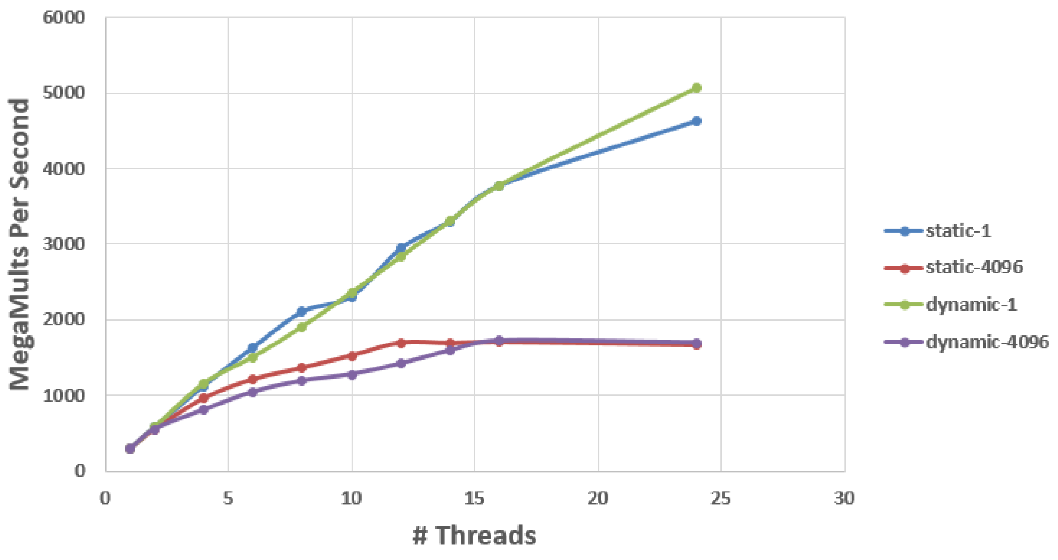

Figure 1. Static and dynamic scheduling with different chunksizes - performance in rabbit

4.

Question: What patterns are you seeing in the speeds?

As you can see in Figure 1, (1) as the threads increases, no matter it is static or dynamic scheduling, the performance increases. But after a certain number of threads, the performance stops increasing, tends to be steady; (2) at both chunksize = 1 and 4096, the static and dynamic scheduling performance are very close (almost the same); (3) when chunksize = 1, the performance is better than when chunksize = 4096, no matter if it's static scheduling or dynamic scheduling.

5.

Question: Why does chunksize 1 vs. 4096 matter like that?

This is caused by the accumulated uneven assignment of the calculation load. The outer for-loop loops through i = 0 ... ARRAYSIZE-1 iterations. For each of these iterations, an inner for-loop loops through j = 0 ... i. When the chunksize = 1, each thread got assigned one iteration and then the assignment start over. In this case, the difference between each assignment is small, although the second iteration may heavier than former one, but it's very small, therefore each threads can be well utilized. As you can see in the figure 1, the performance is better when chunksize = 1.

However, when the chunksize = 4096, each thread got assigned 4096 iteration and then assignment start over. In this case, the 4096 iteration that second thread got is heavier calculation than the 4096 iteration first thread got. When there are a lot of iterations need to be assigned, the accumulated difference between assignments that each thread got will make a difference in the performance. Some of the threads will finish work early and become idle while others are still running, this lead to the poor performance as you can see in figure 1 when chunksize = 4096.

6.

Question: Why does static vs. dynamic matter like this?

In this experiment, The outer for-loop loops through i = 0 ... ARRAYSIZE-1 iterations. For each of these iterations, an inner for-loop loops through j = 0 ... i. That means the work load increases at each iteration. At static scheduling, each thread got a predefined chunksize of iterations, and the assignments start over. Thus, the first thread would always finish the work first since it got the least amount of work load, and then second thread finishes, and the pattern keeps going. In the case that the scheduling is dynamic (”Round-robin assignments”), when a thread from the pool becomes idle, the Master gives it a new assignment. In this case, the first thread should always finish work first, so it got assigned certain chunksize of the iteration, and then second thread finishes, and got assigned certain chunksize of the iteration. This basically follows static scheduling pattern. That’s why the static and dynamic scheduling have the similar performance.

If in an experiment that each iteration’s workload is random, then dynamic scheduling probably does a better job in a large scale calculation.